2013

Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

[TYRE PRINT IDENTIFICATION THROUGH REGISTRATION TECHNIQUES AND ATTRIBUTE GRAPHS]

Author: Pau Poquet Domènech. Advisor: Dr. Francesc Serratosa. 9th September

TABLE OF CONTENTS

1. INTRODUCTION

MOTIVATION

My advisor told me that he had a project in mind. The department found a tire tread database that could be put to good use. Researching about it we found out that there was very little research done in this field and there were probably some other ways to improve them further. The lack of a competent tire recognition method, plus the possibility to work on an image recognition research problem were the key reasons that motivated me to work on the project we are presenting to you.

Tire tread classification problem is primarily of use in two areas. First and foremost, forensics. It could come of great use in a case involving motor vehicles to be able to at least determine what kind of tire has been in a crime scene, and subsequent analysis of the possible vehicles wearing that same tire. Secondly, it could also be used by tire manufacturers to prevent copyright infringement of a tire tread pattern.

We want to think that given a proper use, this technique could be of help to forensic labs around the globe, and perhaps, on some extent, it could also be used on other fields of image recognition.

AIMS & OBJECTIVES

Real world applications are always the highest motivation you can get. Being my first research paper I felt that focusing on a concrete and concise goal was of real help.

We looked for a way to provide the user to identify a tire model, given a partial tire tread image from the ground. At least we need a way to guide him along and choose a few of the extensive database with probable solutions to the tire tread he is looking for and not having them to look for in a three hundred or more tire tread images.

The concept of a partial tire tread image, is the key to this work, and what it makes it unique amongst other systems. Most existing systems work with a complete tire tread, but in most cases, all you can get is a small part of the complete tread. This is why we focused on proposing a solution able to work with just a part of it.

On the other hand this also was also one of the major pitfalls we encountered. Not having the complete information of what we are looking for made all the existing research on tire tread matching unusable, leading us to start from the ground-up.

We are looking for tire tread recognition in its full extent. We believe that any step we take to our final goal is welcome. Given that the user should be supervising the software already he could also be the one deciding which option is the best, given a current result presented in a correct fashion for him to do so. An exact matching would definitely be great but, as in most cases on computer vision field, this is nearly impossible.

ORGANIZATION

The thesis document is organized in 7 chapters.

Chapter 2 consists of an explanation of the required concepts and methods used for our proposed solution and a brief explanation of our application on the project.

Chapter 3 is devoted to describe the current state-of-the-art on the tire tread recognition problem. It consists of an explanation on the methods and decisions which their work was based on, to allow us to compare it to our work and get some ideas.

Chapter 4 explains the tire representations that we tried and how we save the data extracted from each tire.

Chapter 5 is a step-by-step guide to our proposed solution with a few examples for a correct understanding of the method.

Chapter 6 provides a wide array of results with examples of many possible scenarios and a further analysis of them.

Chapter 7 draws conclusions and provides guidelines and ideas for future work and optimization.

2. USED METHODS

HARRIS CORNER DETECTOR

The Harris corner detector is a popular interest point detector due to its strong invariance to: rotation, scale, illumination variation and image noise. The Harris corner detector [1] is based on the local auto-correlation function of a signal; where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions. A discrete predecessor of the Harris detector was presented by Moravec [2]; where the discreteness refers to the shifting of the patches.

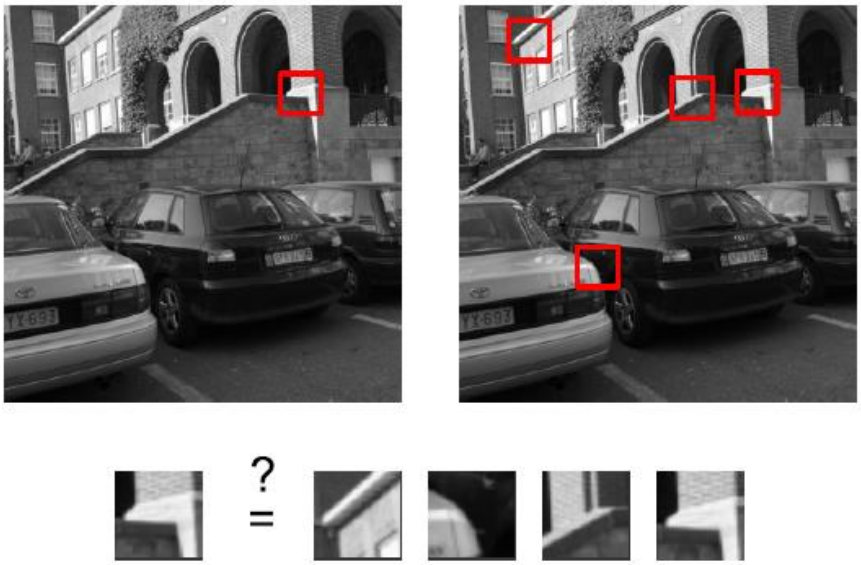


FIGURE 2-1. DISTINCTIVE PATCHES

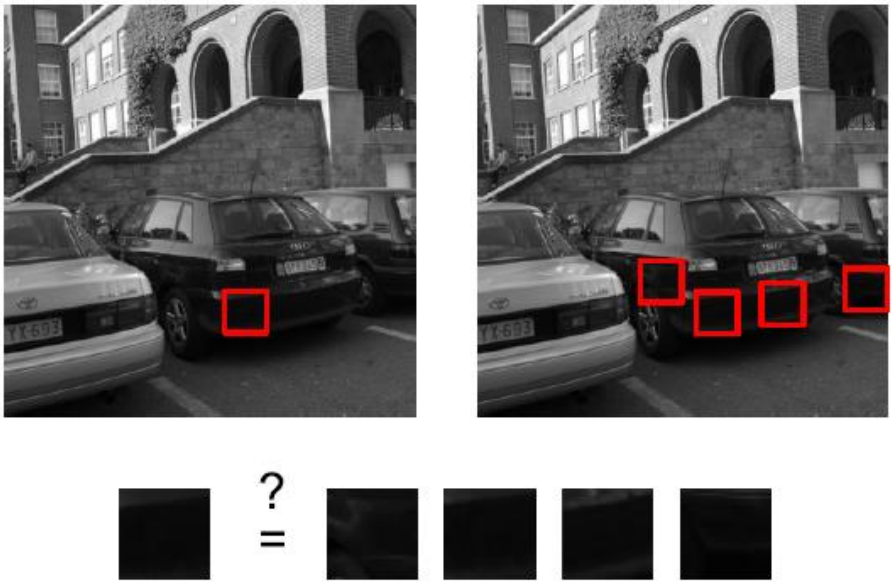


FIGURE 2-2. INDISTINCTIVE PATCHES

Figures 2-1 and 2-2 show images with highlighted distinctive and indistinctive patches to see the difference. The Harris Corner Detector is just a mathematical way of determining which patches produce large variations when moved in any direction. Figure 2-1 highlights patches where the variation is high on a small shift of the patch. These patches are considered corners with an assigned score. On the other side, Figure 2-2, highlights regions that represent small variations of the image given a small shift. Those are not considered important regions; therefore they are given small scores. With each window, a score R is associated. Based on this score, you can figure out which ones are corners and which ones are not. The score also makes it possible to choose among the maximum number of corners needed.

SHI-TOMASI CORNER DETECTOR VARIATION

The Harris detector has a corner selection criteria. A score is calculated for each pixel, and if the score is above a certain value, the pixel is marked as a corner. The score is calculated using two eigenvalues. That is, two eigenvalues of a function are used. The function manipulates them, and returns a score.

Harris method calculates the score R using this function.

$$R = \det M - k(trace\ M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\ M = \lambda_1 + \lambda_2$$

Shi and Tomasi [3] suggested that the function should be done away with. Only the eigenvalues should be used to check if the pixel was a corner or not.

In their paper, Shi and Tomasi demonstrated experimentally that this score criteria was much better. If R is greater than a certain predefined value, it can be marked as a corner.

For **Shi-Tomasi**, it's calculated like this:

$$R = min(\lambda_1, \lambda_2)$$

Although they just adapted the Harris corner detector to evaluate the corner differently, it has been proven to be a better detector than Harris's. Our tests have proven so too.

ITERATIVE CLOSEST POINT

Iterative Closest Point (ICP) [4] is an algorithm employed to minimize the difference between two clouds of points. ICP is often used to reconstruct 2D or 3D surfaces from different scans, to localize robots and achieve optimal path planning (especially when wheel odometry is unreliable due to slippery terrain), to co-register bone models, etc.

The algorithm is conceptually simple and is commonly used in real-time. It iteratively revises the transformation (translation, rotation) needed to minimize the distance between the points of two raw scans.

Inputs: points from two raw scans, initial estimation of the transformation, criteria for stopping the iteration.

Output: refined transformation. (Rotation and Translation Matrices)

On the first step we have a target cloud of points and a source cloud of points.

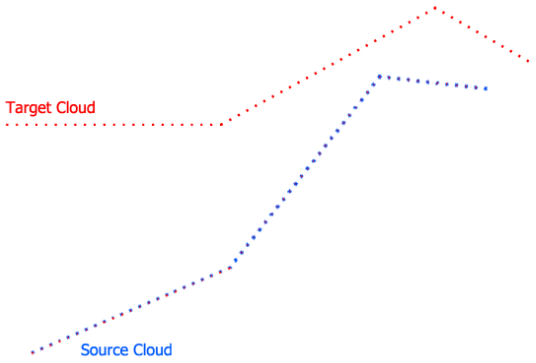


FIGURE 2-3. TARGET/SOURCE CLOUD OF POINTS

Essentially, what we are trying to find is the translation and rotation matrices that allows us to approximate the source cloud of points to the target cloud of points. The next step is to identify which points of the source cloud are just merely transformed/rotated. This is called the correspondence problem.

ICP assumes that the points better suited for the adjustment are the closest points (hence the name)

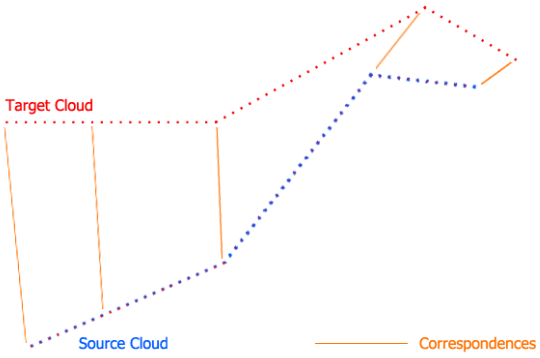


FIGURE 2-4. CORRESPONDENCES BETWEEN TARGET/SOURCE CLOUDS

The decision of which are the closest points is taken using the smallest Euclidean distances.

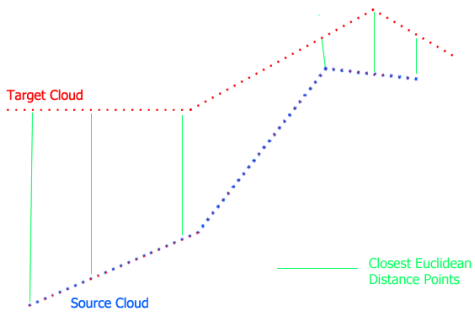


FIGURE 2-5. CLOSEST EUCLIDEAN DISTANCE POINTS

The main goal is to keep minimizing this Euclidean distance until we reach an acceptable alignment. Example of three different iterations of the ICP:

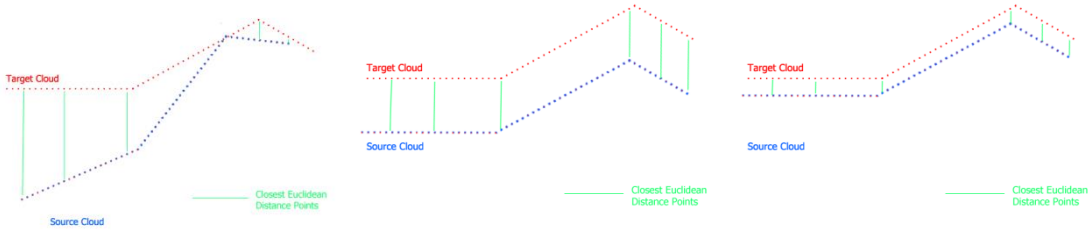


FIGURE 2-6. ICP ITERATIONS

Essentially, every iteration of the ICP does the following:

1. Find closest points: using the Euclidean distance between the target/source point clouds.
2. Calculate alignment: calculate what is the best rotation/translation required to be performed on the source point cloud to align it with the target point cloud.
3. Regenerate source cloud: after applying the transformation (rotation and/or translation) to the source cloud, regenerate the modified point cloud. The visual effect of this is the source cloud either appears closer or further from the target cloud.

The final result is a 4x4 matrix with the necessary translation matrix and rotation matrix to achieve the best transformation.

HUNGARIAN METHOD

The Hungarian method is a combinatorial optimization algorithm which solves the assignment problem in polynomial time and which anticipated later primal-dual methods. It was developed and published by Harold Kuhn in 1955 [5] [6], who gave the name "Hungarian method" because the algorithm was largely based on the earlier works of two Hungarian mathematicians: Dénes Kőnig and Jenő Egerváry.

James Munkres reviewed the algorithm in 1957 and observed that it is (strongly) polynomial. Since then the algorithm has been known also as Kuhn–Munkres algorithm or Munkres assignment algorithm. The time complexity of the original algorithm was O(n^4), however Edmonds and Karp [7], and independently Tomizawa [8] noticed that it can be

modified to achieve an O(n^3) running time. Ford and Fulkerson extended the method to general transportation problems. In 2006, it was discovered that Carl Gustav Jacobi had solved the assignment problem in the 19th century, and the solution had been published posthumously in 1890 in Latin.

The following algorithm applies to a given n × n cost matrix to find an optimal assignment.

1. Subtract the smallest entry in each row from all the entries of its row.
2. Subtract the smallest entry in each column from all the entries of its column.
3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.
4. Test for Optimality:
    a. If the minimum number of covering lines is n, an optimal assignment of zeros is possible and we are finished.
    b. If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.
5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

If we follow an example step-by-step, this is how we solve the assignment problem:

A,B,C have to reach three different destinations (1,2,3) indistinctively. If we have to minimize the costs, which way has each one have to take?

	1	**2**	**3**
A	250	400	350
B	400	600	350
C	200	400	250

Step 1. Subtract 250 from Row 1, 350 from Row 2, and 200 from Row 3.

250	400	350		0	150	100
400	600	**350**	->	50	250	0
200	400	250		0	200	50

Step 2. Substract 0 from Column 1, 150 from Column 2, and 0 from Column 3.

0	**150**	100		0	0	100
50	250	**0**	->	50	100	0
0	200	50		0	50	50

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

~~0~~	~~0~~	~~100~~
~~50~~	~~100~~	~~0~~
0	50	50

Step 4. Since the minimal number of lines is 3, an optimal assignment of zeros is possible and we are finished.

0	**0**	100
50	100	**0**
0	50	50

Since the total cost for this assignment is 0, it must be an optimal assignment.

Here is the same assignment made to the original cost matrix.

	1	**2**	**3**
A	250	**400**	350
B	400	600	**350**
C	**200**	400	250

BIPARTITE GRAPH MATCHING

The bipartite graph matching algorithm [9] [10] solves the error-tolerant graph matching problem and it has lately shown very good performance on several databases. Considering its performance and speed, we consider it is a good algorithm to compute the graph edit distance. The algorithm approximates the graph edit distance (quadratic assignment problem) to the linear assignment problem by using a cost matrix. They define the cost matrix as follows.

$$C = \begin{bmatrix} \overbrace{\begin{matrix} c_{1,1}^{p,q} & c_{1,2}^{p,q} & \cdots & c_{1,M}^{p,q} \\ c_{2,1}^{p,q} & c_{2,2}^{p,q} & \cdots & c_{2,M}^{p,q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N,1}^{p,q} & c_{N,2}^{p,q} & \cdots & c_{N,M}^{p,q} \end{matrix}}^{A} & \overbrace{\begin{matrix} K_n & \infty & \cdots & \infty \\ \infty & K_n & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & K_n \end{matrix}}^{B} \\ \underbrace{\begin{matrix} K_n & \infty & \cdots & \infty \\ \infty & K_n & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & K_n \end{matrix}}_{R} & \underbrace{\begin{matrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{matrix}}_{S} \end{bmatrix}$$

where C denotes the node substitution cost and K denotes the graph edit distance node insertion and deletion. Using this cost matrix, either the Hungarian algorithm or the Munkres algorithm [11] can be used to compute the bijection that minimize the correspondences.

GENERALIZED HOUGH TRANSFORM

The Generalized Hough Transform or GHT, introduced by Dana H. Ballard in 1981 [12], is the modification of the Hough Transform using the principle of template matching. This modification enables the Hough Transform to be used for not only the detection of an object described with an analytic equation (e.g. line, circle, etc.). Instead, it can also be used to detect an arbitrary object described with its model.

The problem of finding the object (described with a model) in an image can be solved by finding the model's position in the image. With the Generalized Hough Transform, the

problem of finding the model's position is transformed to a problem of finding the transformation's parameter that maps the model into the image. As long as we know the value of the transformation's parameter, the position of the model in the image can be determined.

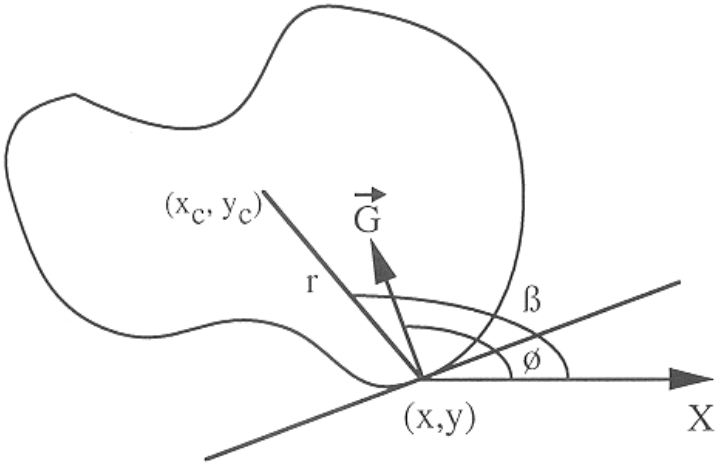


FIGURE 2-7. GENERALISED HOUGH TRANSFORM

The original implementation of the GHT uses edge information to define a mapping from orientation of an edge point to a reference point of the shape. In the case of a binary image where pixels can be either black or white, every black pixel of the image can be a black pixel of the desired pattern thus creating a locus of reference points in the Hough Space. Every pixel of the image votes for its corresponding reference points. The maximum points of the Hough Space indicate possible reference points of the pattern in the image.

Although our use it is slightly different from the general application of this method, it suits our needs.

The main drawbacks of the GHT are its substantial computational and storage requirements that become acute when object orientation and scale have to be considered.

3. STATE-OF-THE-ART

DEFINITION OF BASIC TIRE TERMS

In order to completely understand the following methods it is necessary to have a basic notion of some classic, specific terms in the world of tires [13].

Tire tread: The tread is the whole part of the tire that comes in contact with the road surface. The portion that is in contact with the road at a given instant in time is the contact patch. The tread is a thick rubber, or rubber/composite compound formulated to provide an appropriate level of traction that does not wear away too quickly. Basically there are three main patterns that compose the tire treads: Lug patterns, rib patterns or block patterns.

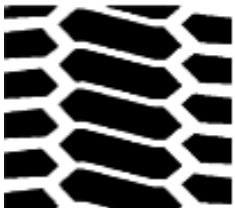


FIGURE 3-1. LUG PATTERN EXAMPLE

Lug pattern: They are oriented primarily in the radial direction and are separated from each other in the circumferential direction by void areas. They generally provide high braking force, and excellent traction on unpaved surfaces.

FIGURE 3-2. RIB PATTERN EXAMPLE

Rib pattern: Rubber islands of the tread pattern that are oriented primarily in the circumferential direction and are separated from each other in the radial direction by void areas. They generally provide low rolling resistance, comfortable ride, good steering and relatively low noise generation.

FIGURE 3-3. BLOCK PATTERN EXAMPLE

Block pattern: In this pattern, the grooves are cut across the tread. Provides outstanding braking force and good traction on snow or in muddy terrain.

Tire grooves: These are the recessed or void parts of the tire tread. Their primary objectives are to allow the water to flow between the parts of contact and to permit the tire to adapt to the ground and the physical deformation from the weight and other forces that inflict.

Having a basic understanding of Tires and its related terms, we are now proceeding to comment existing research papers on which tire tread recognition is in use to some extent.

Back in 2003 D. Colbry et al. [14] published the first paper that approaches to the tire recognition problem we could find. The paper is named 'Pattern Recognition for Classification and Matching of Car Tires'.

This paper was written at a time when the tire tread recognition methods were fully manual. Each year a company named Tire Guides, Inc., publishes a book of tread patterns and also produced software such as the Tread Assistant [15] and Wheel Inspection [16], which had an extensive visual database that can be searched by manufacturer or nominal type (e.g., snow tires, all-terrain tires, high-performance tires). Tread Assistant allowed the user to load a picture of the tire that is to be compared into a separate window within the software. However, there was no automated method for retrieving candidate images and the user must complete the entire tire matching process manually. As of now, we were unable to find neither of these software assistants or webpages. The book of tread patterns [17] by Tire Guides Inc. does still exist and it is being published every year with an updated list of tire treads. 2013 version is already available.

The proposed method aims to refine and identify some tire tread print features to allow them to filter out from an extensive database.

The method works basically as follows. First a preprocessing of the tire images they use:

1. Standardize the images by manually converting them to have the same tire orientation and ensure that the file is loaded as a gray scale image. Crop bordering white space to ensure that the tire is in the same region of each image.

2. Select relevant tread from each tire and crop the rest of the image.

3. Find the edge regions of each tread and convert the pattern to a binary file.

4. Convert the binary image using a two-dimensional fast Fourier transform (2DFFT);

5. Normalize the data by scaling each 2D-FFT to a 64 x 64 pixel image.

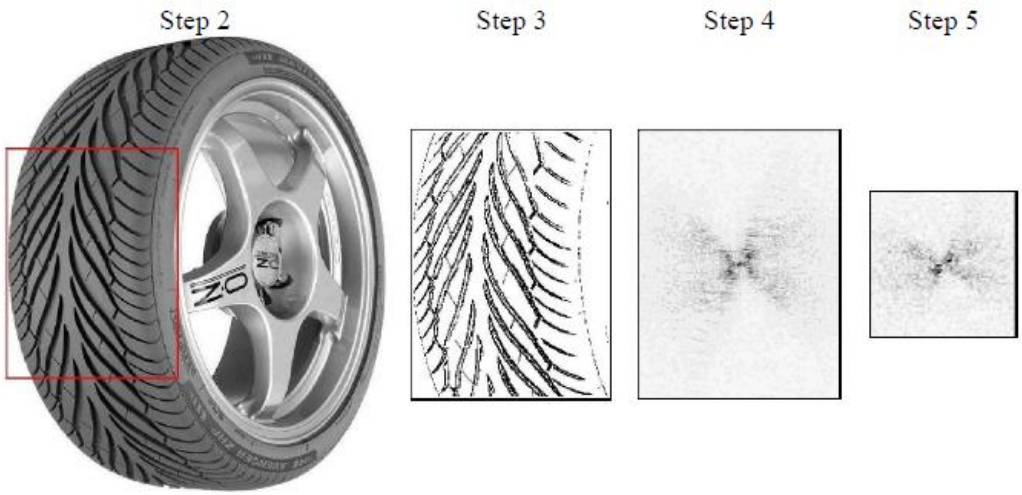


FIGURE 3-4. COMPLETE PRE-PROCESSING STEPS.

Once they get this images processed, they develop a feature extraction algorithm of this own which allows them to separate the images in two dimensions, specifically separating 'rib' and 'lug' features of a tread pattern. The first classification algorithm was developed using PCA (Principal Component Analysis) and PSA (Power Spectral Analysis) with limitations on the performance and the matching rate. Afterwards they tried a prototype using a k-nearest neighbor classifier [18]. The initial prototype based on the knn classifier was able to select similar images from the database to match a test sample and reduced the number of comparisons that must be made to 14.5 in the average case, and 2/3 of the dataset in the worst case. While these error rates are higher than what would generally be considered useful for a traditional pattern recognition problem, for the domain of tire tread matching problems this prototype Tire Matching System does reduce the amount of work required from a human user in order to find a matching pattern.

A second generation of the Tire Matching System was implemented using geometric filters of the features captured by the 2D-FFT. This system is more robust and allows a given tread pattern to be parameterized within the space of rib-and lug-type design features, significantly reducing the amount of images a human user must examine in order to find a matching tread pattern. They analyze the tire further to get additional sub-classes such as '3-rib', '4-rib' or '5-rib' patterns

Both systems did simplify the number of images to be examined but they were using a rather short tire database of 90 images with cases in which they were only able to reduce to 60 the number of images to browse. It certainly was a nice approach given the time in which the paper was written.

RECOGNITION OF TIRE TREAD PATTERNS BASED ON GABOR WAVELETS AND SUPPORT VECTOR MACHINE

Back in 2006, Huang et al. [19] also tried a Tire Tread Pattern Recognition but, this time, based on a whole new distinct approach. They tried the recognition based on Gabor Wavelets [20] and Support Vector Machine [21]. Their technique works as follows; they first preprocess the input images morphologically to enhance the features of the tire surface.

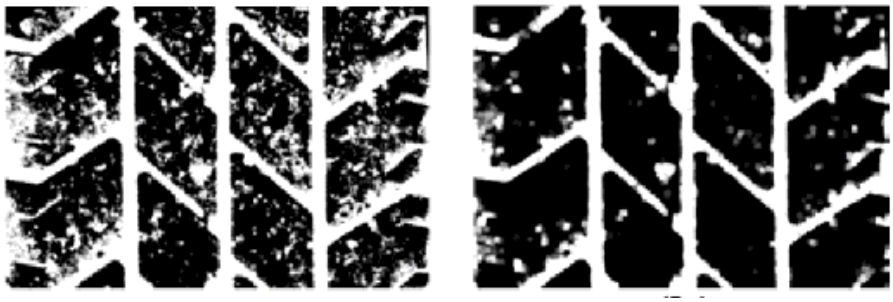


FIGURE 3-5. TIRE PATTERNS HIGHLIGHTED WITH THE PREPROCESSING.

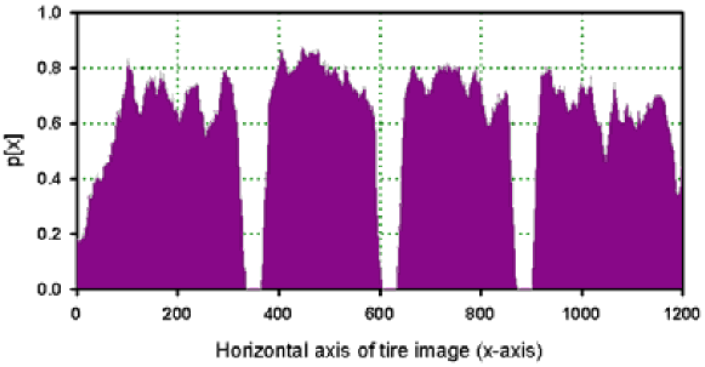


FIGURE 3-6. REPRESENTED GROOVES OF THE TIRES, IN THIS CASE 4

Afterwards they look for the wavy or groovy patterns of the tires and train various support vector machines (SVM) classifiers. The representation of the features is represented by Gabor wavelets and further extraction is achieved by principal component analysis (PCA).

There was another nice touch to the classifiers, while training they provided the same image with a vertical mirrored projection and with a horizontal mirrored projection. This allows for a usable technique whether it is a left tire, right tire and if it is going forward or backwards.

Finally the matching is achieved by the classifiers previously trained in SVM, Euclidean distance and cosine distance. Surprisingly the recognition rate reaches a maximum of 60% with 15 tire treads are used with previously trained classifiers. After reading this paper and all the effort put on trainig we were a little bit disappointed with the recognition rate.

OTHER RESEARCH RELATED TO TIRE PRINT IDENTIFICATION

Some other interesting but not directly related references are deeply related to forensics. Mostly applied on ways to analyze, scan and capture the tire treads from distinct surfaces.

In 2007 [22] published a paper about the three-dimensional documentation of footwear and tire impressions in snow, offering the opportunity to capture additional fine detail for the identification as present photographs. Using a chemical product to highlight the differences in height of the snow and making it a harder surface. Afterwards, they take pictures and cast a ceramic mold over it. They finally manage to create a 3D model of the shoe using different cameras on the crime scene. This example is made using a shoe tread print, but it should work the same on tires.

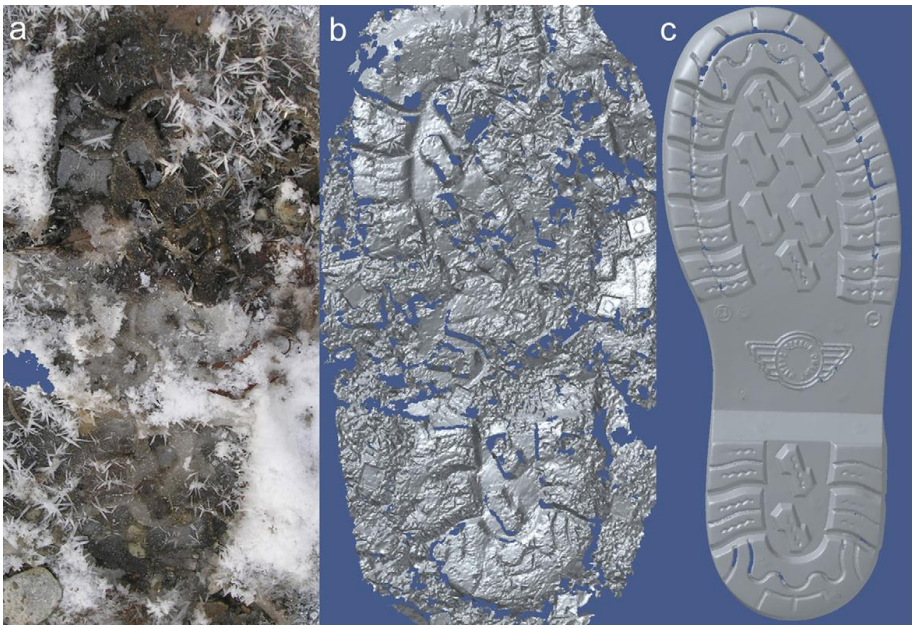


FIGURE 3-7. (A) PHOTOGRAPH OF THE IMPRESSION. (B) THE 3D MODEL OF THE DIGITIZED IMPRESSION DISPLAYS THE RECOVERED PARTIAL DETAILS. (C) RESULT OF THE SURFACE SCANNING OF THE SOLE OF THE SHOE.

Another paper intimately close to forensics [23] is one that is a bit macabre. Using photogrammetry over the head of an injured person they determine if this tread mark is from the suspect car. They match the most significant parts on the injury to a relative on the tread tire itself. It is a use for photogrammetry applied to forensics and tire prints. They use it for other type of injuries too.

FIGURE 3-8. PHOTO SERIES OF THE INJURY, UNDER A DOT MATRIX.

They are both important breakthroughs on forensics, but they mostly come before our intended work.

There are existing commercial solutions. Foster Freeman [24], provides two interesting products related to tire print identification. The first one, Treadmate, is the most comprehensive and complete database of tire prints available. Each vehicle tire record in TreadMate contains the tire's manufacturer, the manufacturer's reference for that tire, the date of its release on to the market, a pictorial image of the tire and a set of pattern feature codes that facilitate search and match operations.

The second is SICAR, a software used to get the matching between the image and the Treadmate database. They kindly sent us a demo of the software with a trial. All the matching is done with a filter and user interaction. For instance, how many grooves are there? What is the direction of the inner groove? What kind of pattern is on the block? And so on…

We couldn't find any Artificial Intelligence on the software, other than a well put database with all the characteristics well detailed. This software is from 2006, and they are currently updating the Treadmate database every year.

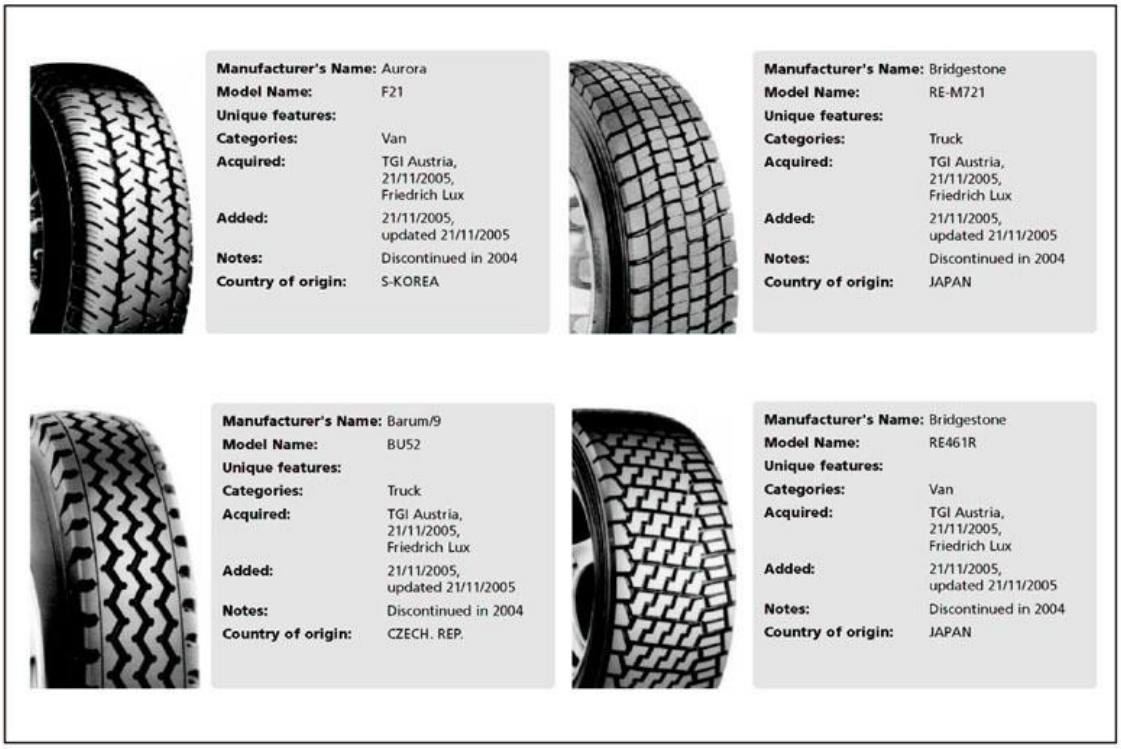


FIGURE 3-9. TREADMATE SCREENSHOT

Having a tire image database encouraged us to work in this regard. This database is composed of about 200 different tire treads with few details about them. These images are 250 x 250 px in size, with squared proportions. Below there is an example of one random image.

FIGURE 4-1. EXAMPLE OF A DATABASE IMAGE

Unfortunately these images suffer from great variations in lighting, and are often lightened from one focal point only provoking uneven situations. This causes that the structure of the tread, even though being repetitive is not always detected as such.

FIGURE 4-2. OTHER EXAMPLES OF TIRE TREADS IN THE DATABASE.

On this figure this lighting variation is clearly visible, using image editors, we try to equalize them and bring the whole tire closer and increment the general contrast.

It seems only obvious that the information you can extract from an image of a tire tread is limited. For starters, color information is completely inexistent and any color information on the picture can be directly discarded. This leaves us with a lot of un-extractable features that can never be used in any way. Texture is out of the picture too, you can barely see any, and even if you can, information did not seem relevant at all.

The first thing that comes to mind when you look at any of these pictures is somehow capturing the tread of the complete tire. To do so, we thought about constructing a graph containing a complete structure of junction points and bifurcations. It might seem easy for us to construct a graph given an image but it certainly isn't so. We created a simple software that helps us extract a graphed structure giving us these results.

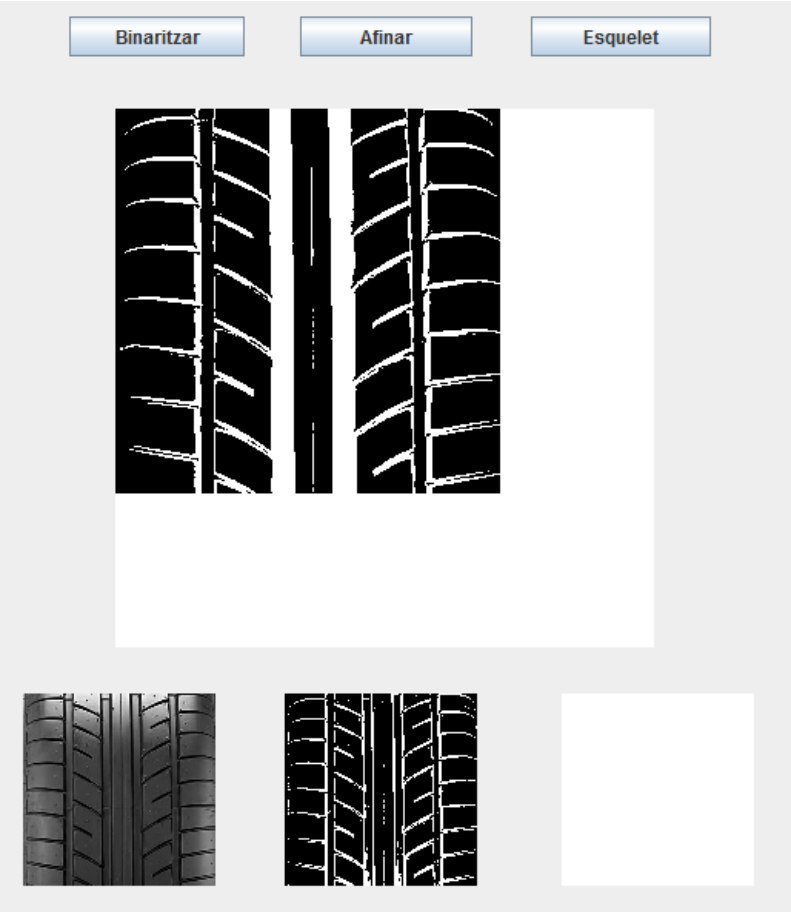


FIGURE 4-3. SCREENSHOT OF THE GRAPH EXTRACTION SOFTWARE

This software allowed us to binarize and manually clean the resulting image. This allows the user to decide whether or not to consider any point white or black, thus simplifying even more whatever part of the image he wants.

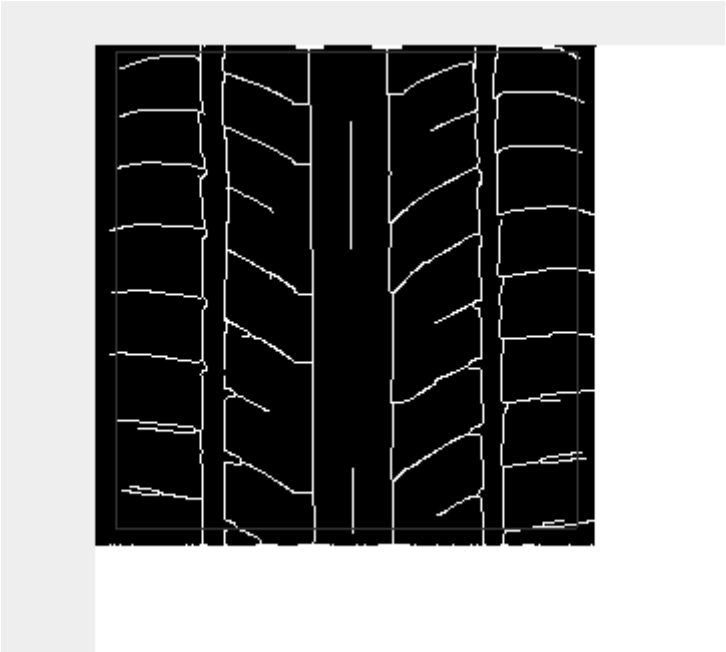


FIGURE 4-4. SCREENSHOT OF THE EXTRACTED SKELETON

Once the clean, binarized image is accepted by the user, the software extracts a simple clean skeleton which contains a simplified line drawing of the structure. This skeleton is the last step on the graph creation, now it is time to classify the endpoints and bifurcations and extract a graph.

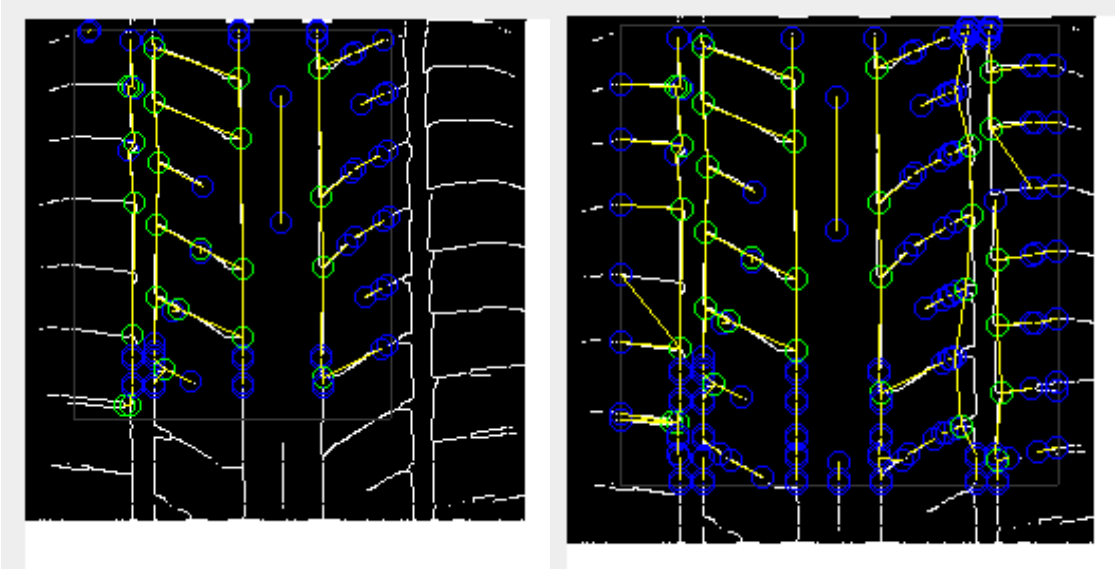


FIGURE 4-5. EXAMPLES OF THE SAME GRAPH WITH DIFFERENT PARAMETERS

We have a variable parameter, either we increase the number of nodes, decreasing the distance between them or we decrease the number of nodes thus increasing the distance among them. We have two kinds of nodes, either a terminal node (blue) or a bifurcation node (green).

Once we have this graph we saved the number of nodes, coordinates (x,y) and the type of node. We also get a connection table between nodes.

We tried a few things with this data including comparison between images, graduated assignment [25] [26] and other methods adapting our data to match. The results were, to say the least, unfortunate. The fact that the graph creation requires human interaction and that the graphs should probably need directionality drove our graph approach virtually unusable. The results were inconsistent and the graph creation, if done right, even with the software took a lot of time. User interaction on the graph creation also presented inconsistencies and differences amongst graphs of the same tire.

As we skipped our graph approach we moved to more traditional image detection methods. Most images are quite similar between them. We tried edge detection methodologies and SIFT (Scale-Invariant Feature Transform) [27] but the feature extraction they provided seemed unlikely to fit in our tire images.

Finally we came along with; perhaps the oldest and simpler of them all, corner detection which seemed to appropriately extract features from the image in a fast and accurate way. The results were not that different from our graphs with just a small fraction of the time and work.

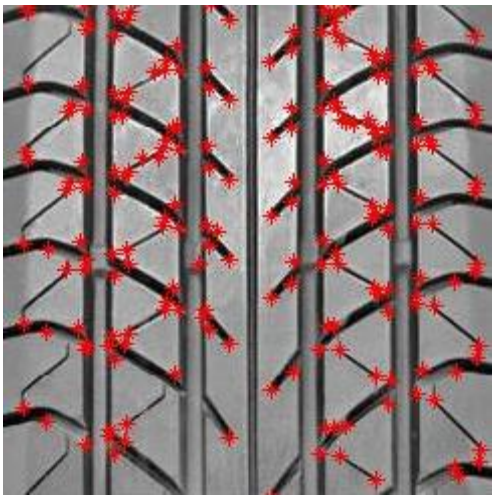


FIGURE 4-6. CORNERS FROM A DATABASE IMAGE.

This method seems to suit our needs, as visible in the Figure 4-6. Corners from a database image.. This information is saved as an (x,y) position points corresponding to each corner found. This information is normally comprised between 200 and 600 corners depending on the complexity of the image. This corner data is used to be compared to our input data. It is worth noting that the better the image contrast and definition of the database is, the better the corners are detected and less outliers stay in the middle of our equation. We concluded that the lightning of the tire is one of the key factors on the corner detection, affecting directly on the corner detection thus making inconsistent results on our database. Perhaps with a commercial database with all the pictures taking with (mostly) the same conditions the results would be more regular.

5. METHODOLOGIES

After all our tryouts we concluded that we will publish only the two most consistent methods amongst the ones we tried.

HUNGARIAN + ICP METHOD

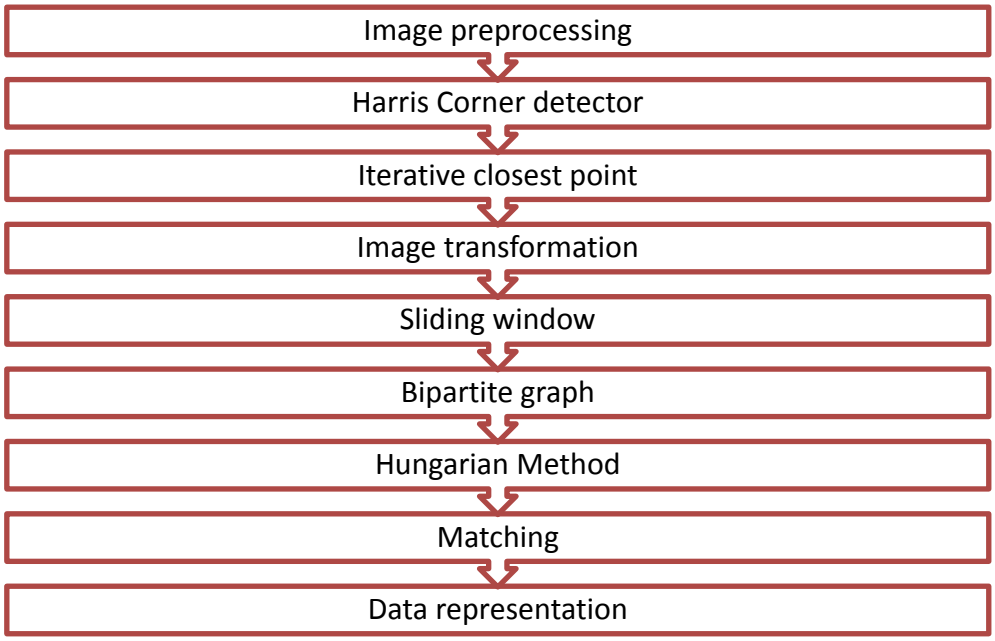


TABLE 1. HUNGARIAN + ICP SCHEMATIC

Image preprocessing: As mentioned before, the first thing we must do, is improve the original images on the database as much as we can. Try to reduce the noise, appliying more contrast, anything that helps us get the best corners we can, given any image.

Harris corner detector: We apply the Harris corner detector (Shi-Tomasi variation) to the database images and the partial tire tread image, crop from now on. The maximum number of corners comes limited by the Density parameter, which we will discuss later. It is worth noting that all the corners on the first two lines of pixels around the images are dismissed because they usually are errors on the image itself. We do not apply the same rule to the cropped image.

Iterative closest point: It is the time for the ICP to work; we compare both datasets (a database image corners with a crop image corners) in order to extract those rotation and translation matrices.

Image transformation: We apply those transformations obtained to the cropped image and we get those images to be as similar as possible in terms of geometrical cohesion.

Although this method does give an approximately good result it was not what we were looking for. For example if there is a repeating pattern in the image (which usually happens), the ICP method gives us the result for approximately the middle of such repeating pattern leaving this result useless. After some tests with simple images we concluded that in order to use this method we should acquire new ways to use it avoiding this behavior and correctly supporting these repeating patterns.

Sliding window: What we did was making the database image smaller by cropping a surrounding area slightly bigger (adjacency parameter) than the input image and move the starting position executing the method multiple times and preventing the middle point problem to happen. As we can see in the image, every step we take it is slightly moved.

FIGURE 5-1. SLIDING WINDOW EXPLANATION

Both parameters (adjacency and offset) are modifiable as inputs. It is worth noting that we don't actually crop the image, instead, we filter out the corners that stay outside of the important area. It is a faster and safer way than actually cropping the image and extracting the corners again, for obvious reasons. This technology leads us to a lot of overhead and unusable results but also provides us with a few good approximations. This is also the reason that provides us with results to return many solutions, selecting the ones that are closer to the input image.

Given that usually, tire treads offer repetitive patterns, also seemed a good way to, perhaps find more than one correct solution or more than one match in the same tire.

Bipartite graph: We take both corner tables (the filtered ones from the database image, and the input image ones with the transformation applied and we generate a new matrix with the Euclidean distances between them. This resulting matrix it is as big as the number of corners of the database image (m) by the number of corners of the input image (n).

Now we construct a bipartite graph, using the previously generated matrix, zeros and infinites and a K (parameter) generating the bipartite graph. This is required for the Hungarian Assignment algorithm to work. It also lets us both dimensions to be different, as the resulting matrix has (m+n by n+m) dimensions, where m is the number of corners of the database image and n is the number of corners of the input image.

2,3	4,4	11,2	1,5	3,2	0,4	K	inf	inf	inf
3,4	0,6	2,4	1,4	5	3,2	inf	K	inf	inf
1,2	3,4	5,6	6	1,1	2	inf	inf	K	inf
2,3	4,1	1,2	0,4	2,2	1,8	inf	inf	inf	K
K	inf	inf	inf	inf	inf	0	0	0	0
inf	K	inf	inf	inf	inf	0	0	0	0
inf	inf	K	inf	inf	inf	0	0	0	0
inf	inf	inf	K	inf	inf	0	0	0	0
inf	inf	inf	inf	K	inf	0	0	0	0
inf	inf	inf	inf	inf	K	0	0	0	0

TABLE 2. EXAMPLE OF A BIPARTITE GRAPH WHERE M=6 AND N=4

The matrix resulting is basically composed of four great areas. The white area contains the distances between original elements. The light gray area represents the cost of erasing these elements, and the dark gray area is the cost of matching new added elements, in our case always zeroes.

Hungarian method: This is where the Hungarian algorithm needs to be applied. K is the only parameter it needs. K is fully customizable and represents the cost of a successful labeling. It also lets us compare two images with a highly different number of corners as the bipartite graph lets us treat it like a square matrix. This is where the assignment becomes a reality.

Once we get the Hungarian method we extract a matrix with a 1 in the position where a match (or a successful labeling) has been considered given the K, and 0 where there is no match. So, every corner with a successful matching on the other image gets a 1. We observed that a decrease in the K extracts more matches, opposed to an increase in the K. Given the size of our images and crops, we determined after numerous tests, that a K=10 seems to be a good fit. Anyhow, this might not always be the right choice, this is why a deeper study of the parameter depending on each image should be made.

Matching: Finally, we took the matches and look for the data it relates. We take then the position plus the offset. We also take the distances, the number of matches and a few more relevant data to decide afterwards what could be the best match. Those were also treated to conclude which data is more relevant for the results to be trusted. We concluded that the best estimation is the summation of the distances divided by the number of matches. Something like the arithmetic mean of distances. This leaves an accurate estimation with just one number of which variation is better, and consequently this is the parameter we compare the others with.

Finally we select the tires which present the least distance, even if there is more than one match on the same tire (usually leading to a repeating pattern).

Data Representation: This is represented with an image of the crop on the top left corner and the eight closest matches with the distance and the position of the crop highlighted on the database image. This lets the user discern which is more appropriate. All the rest of data can also be consulted on a complete table and we are also able to increase the number of matches to show if necessary.

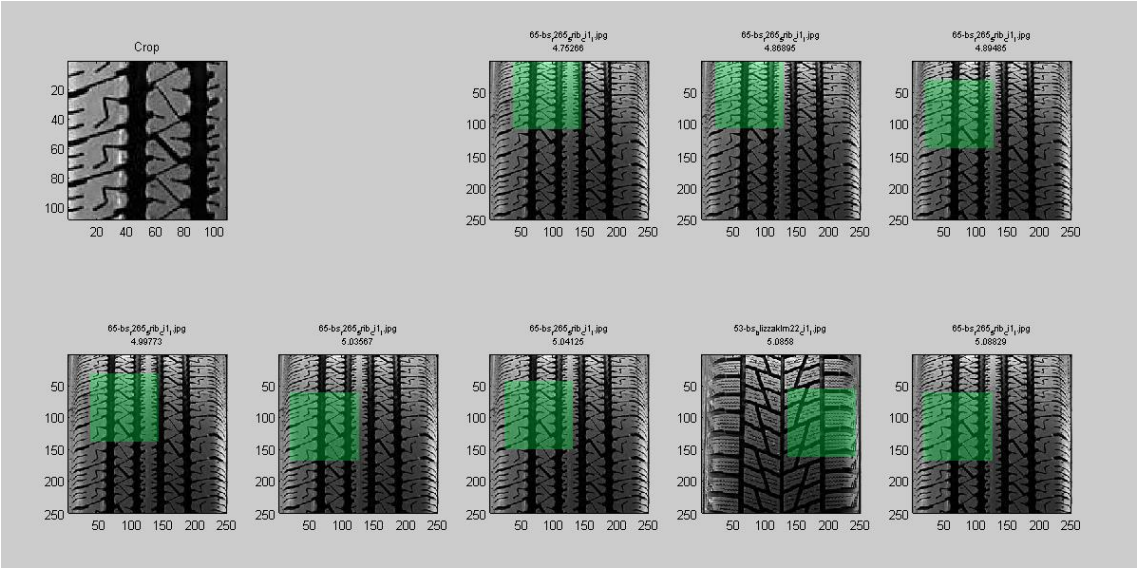
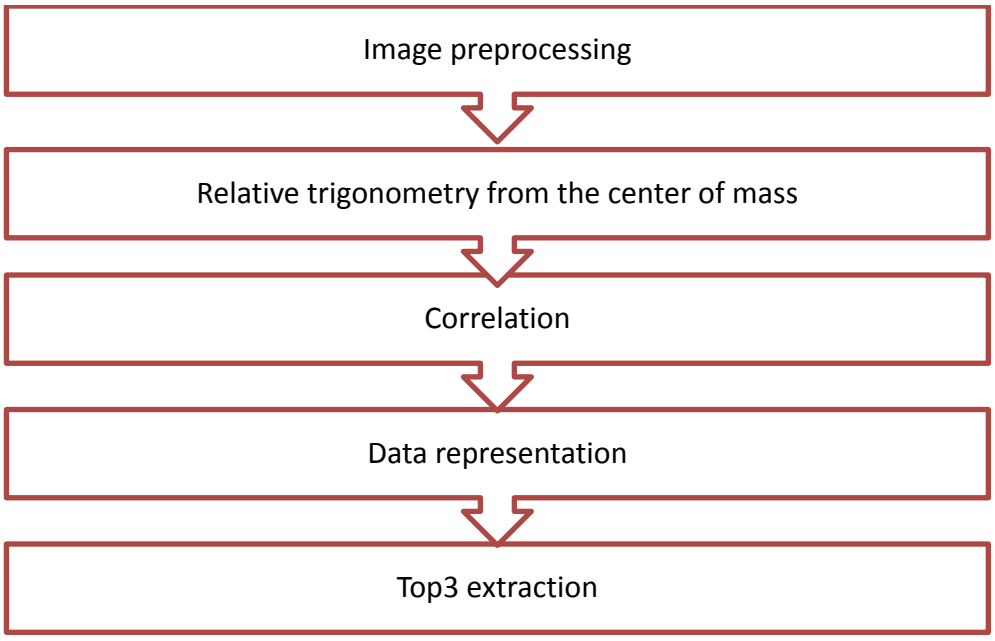


FIGURE 5-2. SCREENSHOT OF THE RESULTS.

GENERALIZED HOUGH TRANSFORM



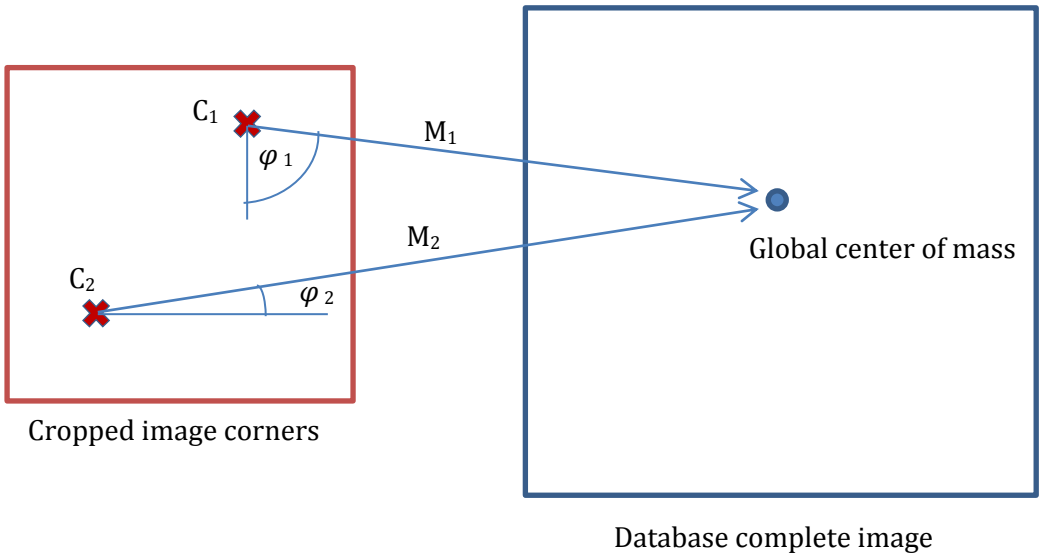
The other method we selected is the Generalized Hough Transform. In order to apply this method to our problem we use the same information (a database of images and a crop from an image to analyze).

Image preprocessing: As on the previous methodology, the first thing we must do is improve the original images on the database as much as we can. Try to reduce the noise, applying more contrast, anything that helps us get the best corners we can, given any image.

Relative trigonometry from the center of mass: Once we have all the corners, we calculate the center of mass of the complete image corners. Now we fill up a table containing for each corner of the partial image the distance and direction needed to reach the global image center of mass. Using basic trigonometry we save the angle and the module of each corner.

$$M_i = \sqrt{\left(X_c - C_{(x)i}\right)^2 + \left(Y_c - C_{(y)i}\right)^2}$$

$$\varphi_i = \tan^{-1}\left(\left(Y_c - C_{(y)i}\right), \left(X_c - C_{(x)i}\right)\right)$$



Where M_i is the module of each corner relative to the center of mass and φ_i is the angle to reach the center of mass. X_c and Y_c are the coordinates of the center of mass from the original image and C_i are the x and y coordinates of each corner.

Correlation: Now that we have an initial relation between both images we need to find out which of the cropped image corners correspond to corners on the original image. We are not only comparing with the center of mass. We are individually comparing each database corner against each vector, evaluating if both corners were exactly the same, where would it be on the complete image. We fill a table with the x (T_x) and y (T_y) positions relating these corners. These positions are calculated with these formulae:

$$T_{(i,j)}^{\ x} = (M_i * \cos\varphi_i) + C_{(x)j}$$

$$T_{(i,j)}^{\ y} = (M_i * \sin\varphi_i) + C_{(y)j}$$

There will be positions (x,y) where more than one corner is supposedly located. Using a matrix P filled initially with zeroes with the same size as the image (one position for pixel) we do the following:

$$P\left(T_{(i,j)}^{x}, T_{(i,j)}^{y}\right) = P\left(T_{(i,j)}^{x}, T_{(i,j)}^{y}\right) + 1$$

This way we keep track of which positions have more than one occurrence, leading us to the most probable region where the crop is located. In the following case Figure 5-3, most positions do not have a match. The region on the left seems to contain more hits reaching a maximum of three points located exactly on the same position.

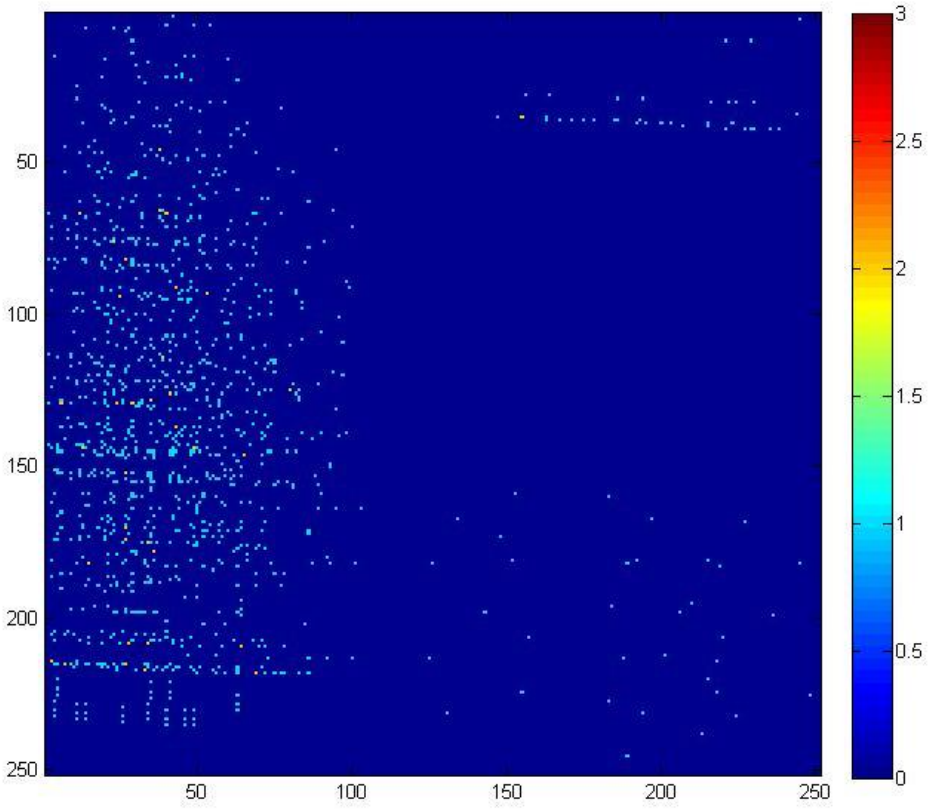


FIGURE 5-3. SCARCE OCURRENCE MATRIX

Data representation: To evaluate the results in a visual manner and have a more accurate idea of where the supposed crop position is, we discretize the matrix P with a modifiable parameter (discretization: how many pixels every matrix position is). We take the resulting positions and see where in the discretized matrix it belongs. This way we are able to look at regions, not exact positions.

To minimize the error, we opted for a point distribution strategy. We add 4 points to the exact position. The four sides of the corresponding point get 2 points and the four resting diagonals get 1 point. This way we can minimize the errors produced by the discretization of the functions.

1	2	1
2	4	2
1	2	1

TABLE 3. POINT DISTRIBUTION STRUCTURE

Top 3 extraction: Once the polling and the calculations are done we decided to keep the three highest polled positions to present the user and also a visual representation with colors of the discretized matrix. These three most polled positions are the ones in which it is more likely that we have a match.

Every position has a number of points (depending on how many matches) and a distance value.

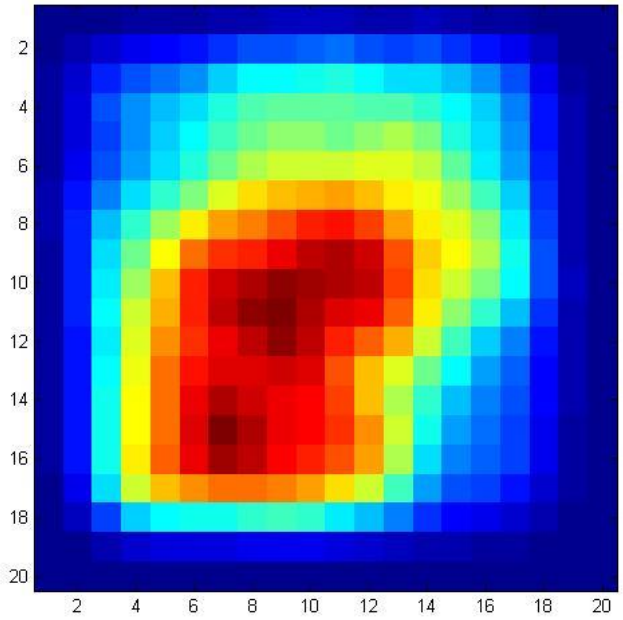


FIGURE 5-4. POSITIONING SCREENSHOT OF THE SECOND METHOD

In this case there are two positions clearly visible as candidates (the ones with the darker red).

Finally, we revert the discretization and we save the three minimum distances to a general table to compare these distances with other database tires for later use. We do not keep the number of hits that each position as they do not seem to have major importance for subsequent analysis.

6. EXPERIMENTAL RESULTS

First of all, we must say that all experiments and tryouts were made using Matlab software [28]. The first set of results is from the first method, the one with the Hungarian method + icp. We selected 15 random images from the complete database we already had. These will be our test bed. We made some other tests to compare it to a real world scenario but all of the rest will be using this 15 image-set. Any other case will be specified.

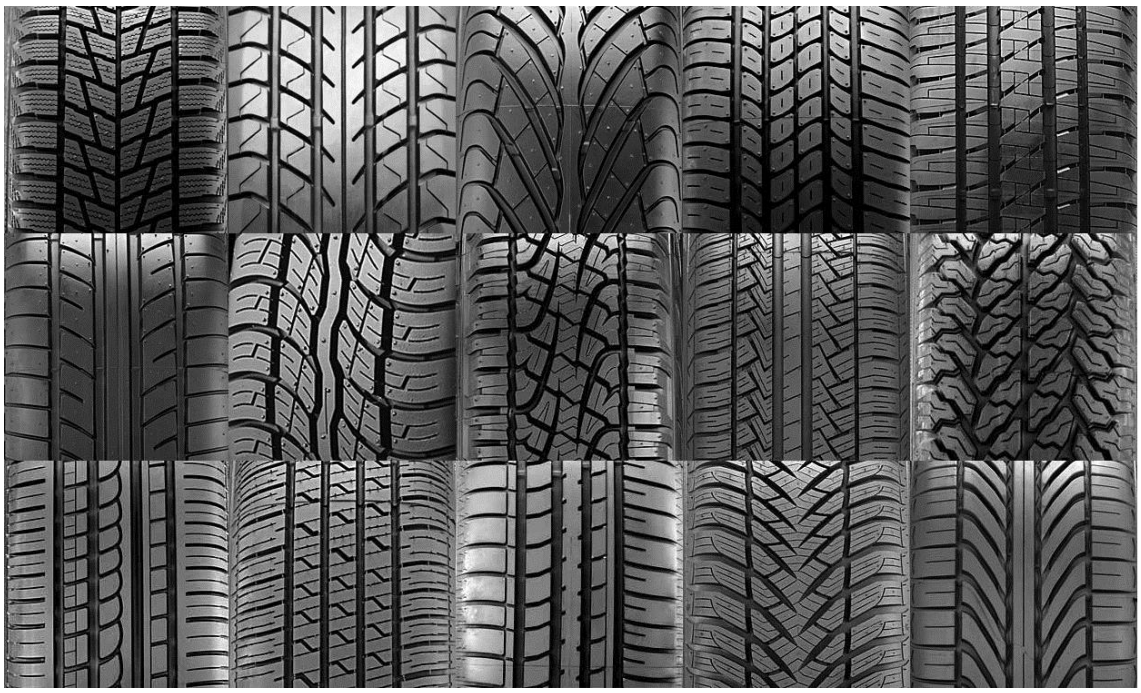


FIGURE 6-1. 15 IMAGES SELECTED AS OUR TESTBED.

To test the recognition rate of the methodology, we paired our dataset to 15 crops, one of each image, in a way that there is supposed to be a right match from our database. Given these crops we pit each of the 15 crops against each of the 15 database images and we extract a table (15 x 15) where we save the smallest Euclidean distance of each crop to the database. This table is very complete and has limited interest for each of our tests, so we have selected two results as important, If we get a match M (in percentage) and if we get in the top three selected matches T3 (in percentage). This way we can offer easy and understandable results.

FIRST METHODOLOGY TEST

PARAMETERS OF THE TEST

As we discussed before there is an important parameterization in each methodology. After a few tests we decided that we are going to use the following parameters for the test:

K=10; **Offsets**=20; **Adjacency**=1.3; **Density**=4;

Where **K** is the cost of a correct graph edit node insertion.

Offsets (in pixels) is the size of the step we take on the sliding window. Obviously the smaller the offset, the bigger the correctness of the match but the longer it takes to execute. This parameter must be in conjunction with the adjacency parameter, the size of

the crop, and the size of the database image. It could be easily transformed to percentage if the sizes are really variable.

Adjacency (proportional) is how much of the size of the crop we take into account at the time of evaluating the corners of a crop. An adjacency parameter of 1.3 means that we increase the area of affecting corners of each sliding window a 30%.

Density (proportional) is how many maximum corners we allow to be in an image. This is a proportional feat that allows us to reduce the number of corners for improved performance. It is just a top limit, which the image should not pass. For instance, if we have a density of 1, all corners should be detected, and therefore all accounted. The **maximum number of corners is**:

$$\frac{image\ width\ (px) * image\ height\ (px)}{2} * \frac{1}{Density}$$

It is not just plain elimination, it takes less corners but it takes the best ones (the ones with highest change) allowing us to increase the performance while keeping almost the same match rate.

The crops were taken from the database, with a square shape and with 106 x 106 px in size. Without filtering, the results table is as following:

7,588	6,690	7,675	5,459	7,103	7,049	6,660	7,671	6,260	7,305	7,053	6,900	6,063	6,948	7,543
7,199	0,581	6,903	7,181	6,413	7,317	6,157	7,260	6,861	7,435	7,990	7,516	7,434	7,721	7,255
7,615	7,180	3,005	6,381	6,837	6,929	7,447	7,684	6,020	6,357	7,140	6,543	7,625	7,788	6,745
6,652	6,481	8,077	4,346	6,121	6,386	6,759	7,751	6,874	6,469	6,129	6,107	6,892	7,091	7,202
6,446	5,140	7,532	6,147	2,408	7,116	6,891	7,894	6,292	7,428	7,283	6,716	6,890	6,942	6,990
6,817	6,592	7,682	6,100	6,806	2,476	6,516	8,129	6,813	6,442	7,169	6,524	6,124	7,552	7,342
7,193	6,913	6,778	6,196	6,762	7,557	1,137	7,248	6,722	5,929	6,546	7,501	7,087	7,524	6,314
6,569	6,559	6,156	6,737	5,560	6,948	7,503	3,135	6,580	6,770	6,787	5,857	7,962	7,727	6,613
7,125	6,724	7,876	6,384	6,836	7,164	7,686	7,537	2,998	5,599	6,982	6,745	6,436	6,046	7,587
6,615	6,914	6,451	6,282	6,856	6,812	7,506	8,276	6,257	4,873	6,889	7,328	5,783	7,396	7,613
7,009	7,273	7,489	7,507	7,296	7,601	7,748	7,594	7,386	6,507	1,892	7,825	7,220	6,496	6,202
6,312	5,850	6,997	5,499	6,001	6,083	5,725	8,172	5,969	6,876	6,510	1,297	7,386	6,372	5,757
5,170	7,373	7,878	7,223	6,103	7,314	8,060	7,867	7,773	6,899	6,836	7,245	0,966	7,536	6,982
7,442	6,310	6,163	4,882	6,365	6,201	6,188	7,680	6,691	6,440	6,712	5,904	7,437	1,778	6,913
7,041	6,621	8,002	6,864	5,403	6,856	6,318	7,637	6,796	6,300	6,956	6,942	7,019	7,509	2,429

TABLE 4. COMPLETE RESULTS TABLE

A plain 15-by-15 table with the smallest distance of each crop with each corner. Ideally, the diagonal should always be the smallest distance, unfortunately this is not always the case. We selected only the top three because our test bed is only of 15 images. If we had selected more images, the top 8, or top 10 could also be optimal selections.

The approximate elapsed time to put 15 crops against 15 database images is around 13 minutes per execution.

Test 1, crops without any modification

% Matches = 93,3 %

% Top 3 matches = 93, 3 %

As it is the first case we would like to show a couple of screenshots of the visual matching with this case:

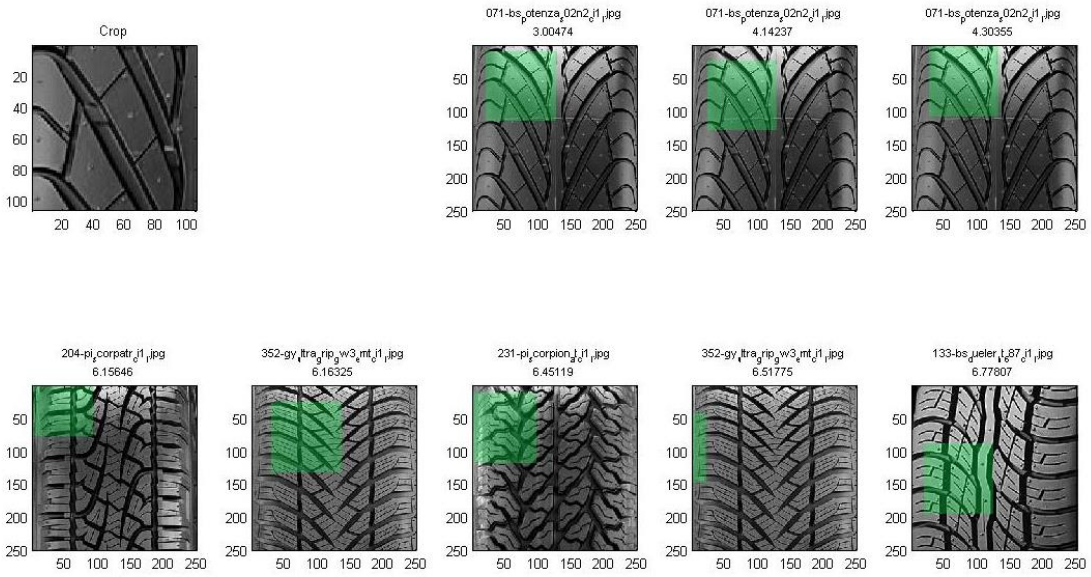


FIGURE 6-2. SCREENSHOT OF A VERY SUCCESFUL TEST

The results, on the Figure 6-2. Screenshot of a very succesful test, are absolutely beautiful. It is, I must add, one of the best scenarios we encountered. The so called crop is visible on the top-left corner. The highlighted green transparent areas are possible solutions proposed by the system. The name of the file is on the top of each image, as is the distance. As you can see the first three hits are from the same tire, also the right one. It is possible for the system to show you the absolute best, like in this case, even if it means repeating the same tire more than once. The other possibility is to just show you one copy of each different tire, which would be more appropriate for cases with more tires. It would also be pretty easy to show you the next 8, in case the system did not select any right one on the first 8 selections.

In the next case, the behavior of the system is more erratic, although also valid for our tests. We observe two correct matches contained on the 8 suggestions.

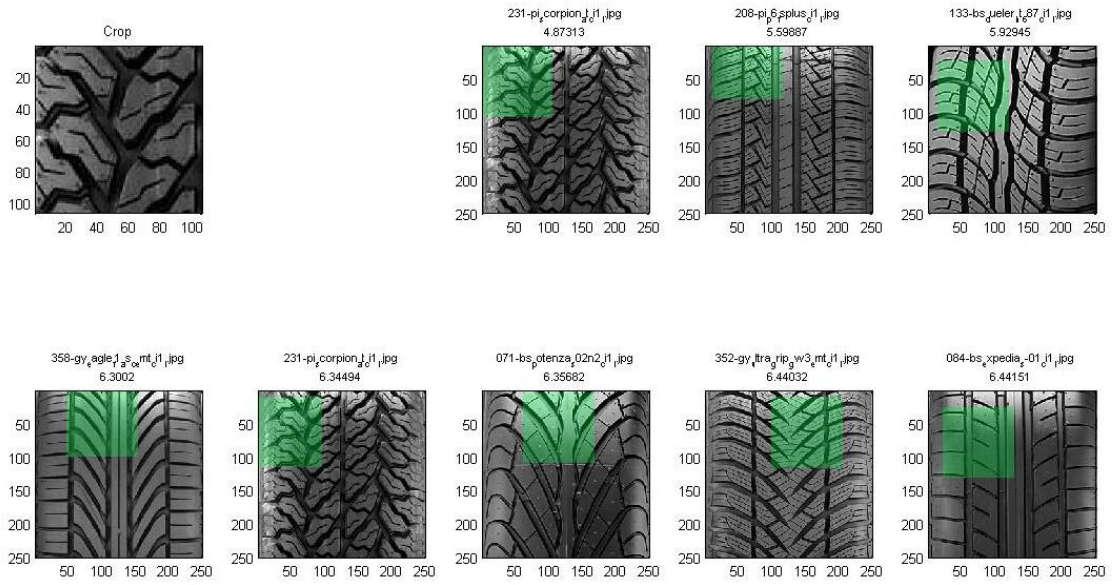
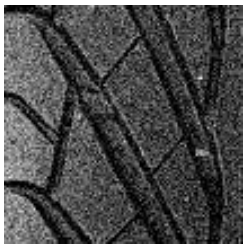


FIGURE 6-3. SCREENSHOT OF A SUCCESFUL TEST.

Test 2, crops with light noise applied, Gaussian blur 12.5 %

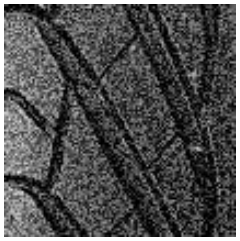


% Matches = 40 %

% Top 3 matches = 53, 3 %

Appreciable decrease in performance, comes with an important reduction of clearness in the crop.

Test 3, crops with heavy noise applied, Gaussian blur 25 %

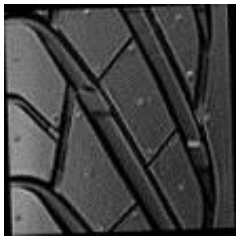


% Matches = 33.3 %

% Top 3 matches = 53, 3 %

Even with the increment in noise the performance is not much worse than in the previous low noise case.

Test 4, crops with light rotation, 1 degree counter-clockwise

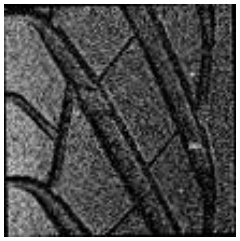


% Matches = 40 %

% Top 3 matches = 66,6 %

It seems that the low rotation is better supported thanks to the icp.

Test 5, crops with light rotation, 1 degree counter-clockwise and light noise applied, Gaussian blur 12.5 %

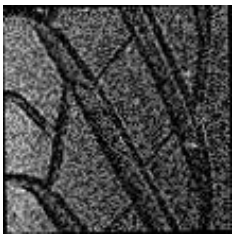


% Matches = 20 %

% Top 3 matches = 46,6 %

These results affect performance in a really considerable way, even if we still have these 46,6%.

Test 6, crops with light rotation, 1 degree counter-clockwise and heavy noise applied, Gaussian blur 25 %



% Matches = 13.3 %

% Top 3 matches = 26,6 %

Under poor circumstances performance and matching rate are just unusable.

Given the circumstances, we believe that rotation can be mostly corrected with a good use of imaging tools. Noise is far more difficult to fix, so we might say it is the biggest problem this methodology could be facing. We concluded that to get the most of this it is of capital importance getting good images of the crop, and the database. The better it looks, the better it works.

As a curiosity and use case experiment we also tried pitting one crop against 100 database images. The time it took to compare was about 17 minutes / execution which is from our point of view quick enough given that this is not a system supposed to work lots of data every day. The results are a little bit of a mixed bag. With 10 different crops against a 100 image database. 6 crops were in the top 10 selections, and the other four on the rest top 25. The best we got was a 4th position in one of them.

SECOND METHODOLOGY TEST

To assimilate and compare both methodologies as much as we can we kept the same datasets and almost the same parameters. All shared parameters are exactly the same.

Density=4; Discretization=3;

Density is kept the same as before. Discretization (in pixels) is the parameter which renders the table in parts. Every 3 pixels we consider it is a region so it is accounted as the same.

Test 1, crops without any modification

% Matches = 60 %

% Top 3 matches = 40 %

As it is clearly visible according to the numbers this methodology is far worse in matching rate than before. The complete table directly under the same experiment as before is as follows:

82,53	67,31	88,25	80,09	76,13	85,10	57,55	100,16	64,21	48,99	66,73	74,50	90,73	60,05	88,56
116,78	8,86	29,30	116,89	0,71	29,91	90,83	52,31	119,63	18,67	26,62	51,21	19,30	86,67	102,15
23,51	94,62	99,11	116,35	44,57	33,59	112,39	67,80	73,45	95,94	83,86	66,77	59,60	51,73	27,06
46,29	55,10	50,32	26,95	52,37	34,47	58,84	35,92	78,58	96,59	74,38	51,79	73,81	63,74	104,18
43,06	91,98	33,23	77,54	6,04	29,30	62,41	67,18	65,13	63,97	53,45	62,02	17,51	50,32	22,73
53,50	45,04	45,30	50,62	50,92	34,47	35,92	110,96	48,50	27,29	44,97	39,43	29,50	36,17	36,91
40,33	23,63	42,29	26,50	26,62	15,70	22,19	38,50	15,89	32,32	18,51	22,73	24,38	64,90	52,54
70,45	97,58	87,54	94,62	81,69	90,00	96,28	93,50	104,99	71,54	80,02	65,18	75,02	103,55	96,28
50,92	42,29	77,81	45,04	48,13	52,37	35,59	65,91	60,60	57,03	48,50	34,47	63,74	50,56	62,41
52,14	71,54	59,50	41,50	38,89	45,30	28,89	78,04	37,64	27,72	44,50	29,30	33,95	102,00	29,50
123,55	130,10	167,50	128,31	180,35	184,61	199,57	140,61	178,39	144,96	179,62	198,39	150,64	161,54	153,50
69,33	60,85	82,05	60,85	54,45	46,74	63,74	60,05	63,50	43,62	46,29	26,95	33,50	49,24	39,65
30,01	37,48	23,51	33,50	21,22	40,70	19,30	27,61	19,30	23,51	14,51	12,51	3,54	38,89	17,68
56,29	70,87	61,14	55,10	56,93	73,57	66,73	78,04	85,28	55,10	71,84	49,24	98,35	69,11	39,65
149,30	100,19	101,32	102,97	143,63	109,44	121,79	108,39	123,84	137,70	137,98	139,82	142,45	107,81	107,81

TABLE 5 - RESULT TABLE WITH THE SECOND METHODOLOGY

Results are much more variable and the distances are far higher. Given the discretization, less precision on the distance is to be found. Sometimes even if these values are chosen given the three best positions, the corresponding value is extremely high.

The results of a successful matching case, in our table crop number 2 are like this:

Dist1 = 37.4767; Dist2 = 8.8600; Dist3 = 10.1242;

N1 = 316; N2 = 291; N3 = 283;

Where Dist1 is the distance to the most voted point and N1 are the number of hits of that position. Dist2 is the distance to the second most voted point (in this case, also the lowest), and N2 the number of hits on that point, and so on.

Let's focus on the successful second tire set. We have this crop and this image, and the resulting graphic.

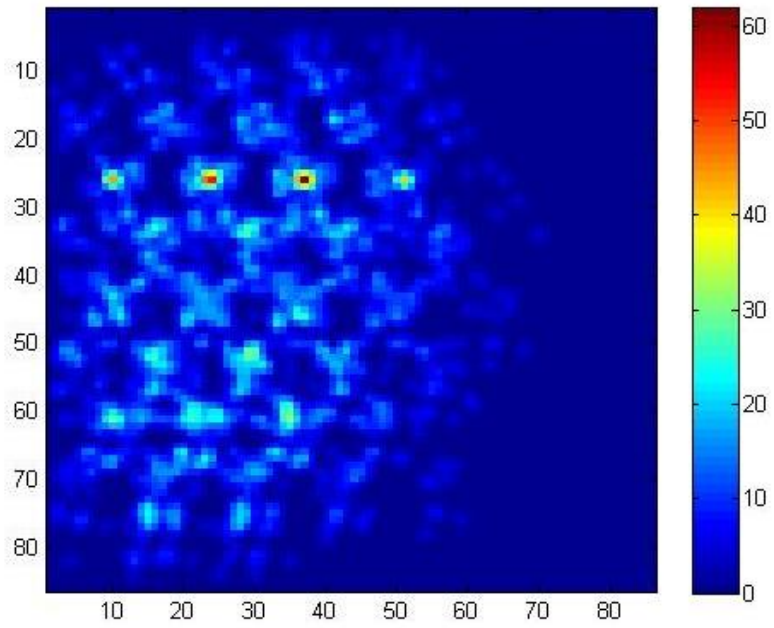
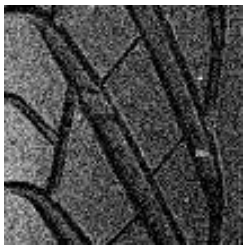


FIGURE 6-4 - POINT DISTRIBUTION ON A SUCCESFUL CASE

As visible the pattern is repeating and the second point, the one in the position (26,26) is the one corresponding to the real position of the crop.

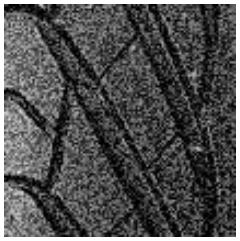


Test 2, crops with light noise applied, Gaussian blur 12.5 %

% Matches = 33.3 %

% Top 3 matches = 40 %

Appreciable decrease in performance, important reduction of clearness in the crop.

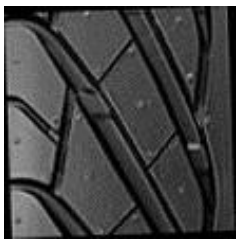


Test 3, crops with heavy noise applied, Gaussian blur 25 %

% Matches = 20 %

% Top 3 matches = 33, 3 %

Even with the increment in noise the performance is not much worse than in the previous low noise case.



Test 4, crops with light rotation, 1 degree counter-clockwise

% Matches = 13.3 %

% Top 3 matches = 20 %

It seems that the low rotation is the worse you can do to this method. ICP on the first methodology seems to improve a lot in this field.

The rest of the tests are not being published because their results are very poor in matching rate. These results are not as good as what we achieved with the first methodology. The only thing in which it drastically improves the first methodology is on speed. A complete test takes about 70 seconds, compared to the 13 minutes of the first methodology.

This second approach could be of use in locating an area or object within an image, but it is not the best choice on the recognition problem that occupies us.

Given the results achieved on previous tire tread recognition papers, our results on the first methodology are not that bad. It is not, by any means, what we were thinking of at the beginning of the project. After months of tryouts and research this is all that we could achieve.

7. CONCLUSIONS AND FUTURE WORK

This Master thesis studies in depth a practical problem and presents several solutions to it. The core of these solutions is not to develop a specific method but to analyze the usefulness of different methods extracted from different research fields and find the way to put them together to solve the problem at hand.

We struggled with a lot of methodologies with arbitrary results and could not find an optimal solution to this problem. Based on what we knew of fingerprint recognition applied to this problem and adapting it to the case. Sadly, all we did was hitting walls.

Perhaps one of the main reasons this field of computer vision has not been deeply researched has something to do with the fact that it is not a simple problem, but a deeply complex problem with serious difficulties in its recognition.

We tried both methodologies on regular images in which we would like to find an object on the images and the results were far better. Both methodologies gave us very convincing results. The repeating and complex patterns a tire tread produces a big increase in the difficulty of a correct recognition.

We are still wondering what should be the best approach to this problem but, unfortunately we weren't able to find it.

As my first research project I learned that research is way more difficult that it seems. You are always walking an uncertain path, reaching for the unknown. You have to learn what tools you have, how you can apply this knowledge and find new and innovative ways to do it. If after all these meticulous steps you do find a good solution it must be greatly rewarding.

However, this has been an enriching personal project. I learned a lot of Computer Vision techniques and algorithms, we had to overcome pitfalls and find solutions. I had to research on existing methodologies. My tutor and lab partners encouraged and helped me a lot.

FUTURE WORK

From our point of view with optimization on various levels this could reach another stage of maturity, and subsequent improvements that we propose below.

GRAPH LEVEL OPTIMIZATION

We still believe that Graphs could bring to the table major improvements on matching. As always the main pitfall is the time consumed on creating a consistent and certain graph and the time consumed also on the graph matching itself.

SOFTWARE LEVEL OPTIMIZATION

Obviously there is a lot of room for software-wise improvement. For starters, the whole software is implemented using Matlab. Even though it is a magnificent software and has permitted us to work easily and run most of the tests in no time. The imaging tools on the

software were also a key factor. It also provided us with a platform with a lot of implemented and optimized methods, like some of the ones we use.

On the other hand the performance is suboptimal. It uses and loads lots of libraries that are useless once the software is programmed and tested.

We propose a C implementation of the core algorithm in which we forecast a significant increase in productivity and performance. We also propose a GUI implementation for the application to be more user-friendly.

TIRE TREAD DATABASE OPTIMIZATION

A prior classification of the tire database could spare a lot of processing time. Perhaps a method similar to the used by the prior references, splitting them into categories like '2-rib', '3-rib' and such. Afterwards we could filter out the affected category giving a significant performance boost and a higher degree of certainty.

The tire tread database could also be more exhaustive and we should be able to get more data of each tire, like years of fabrication or default car models that use them.

Finally, the quality of database images can be really a huge factor on these methodologies results. We believe that if we could have gotten the professional Treadmate database or similar, we would be closer to a satisfactory solution.

PARAMETER STUDY OPTIMIZATION

Deeper study of the parameterization could also be presented. This could provide speed optimization of the moving window, depending on the input image. We could also add a K modifier depending on the tire structure, which is directly related to the distances present in the Hungarian matrix. The sizes, adjacencies and offsets of each execution change radically the results. Those are just a few ideas but when you have 4 direct parameters it is of vital importance a correct adjustment of those to have the best performance possible on the method.

On the Generalised Hough Transform, the parameters of discretization also present variations on the results, although not that important.

8. REFERENCES

[1] C. Harris and M.J. Stephens. A combined corner and edge detector. In Alvey Vision Conference, pages 147–152, 1988.

[2] Moravec, H, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Tech Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute, September 1980.

[3] J. Shi and C. Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), Seattle, June 1994.

[4] T. Jost, H. Ḧugeli: Fast ICP Algorithms for Shape Registration , Lecture Notes in Computer Science Volume 2449, 2002.

[5] Kuhn, H. W. (1955), The Hungarian method for the assignment problem. Naval Research Logistics, 2: 83–97. doi: 10.1002/nav.3800020109.

[6] Kuhn, H. W. (1956), Variants of the hungarian method for assignment problems. Naval Research Logistics, 3: 253–258. doi: 10.1002/nav.3800030404.

[7] Jack Edmonds , Richard M. Karp, Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, Journal of the ACM (JACM), v.19 n.2, p.248-264, April 1972 [doi>10.1145/321694.321699]

[8] N. Tomizawa. On some techniques useful for solution of transportation network problems. Networks, 1:173–194, 1971

[9] Riesen, K., M. Neuhaus, et al. (2007). Bipartite Graph Matching for Computing the Edit Distance of Graphs. GRAPH-BASED REPRESENTATIONS IN PATTERN RECOGNITION. 4538/2007

[10] Riesen, K. and H. Bunke (2009). "Approximate graph edit distance computation by means of bipartite graph matching." Image and Vision Computing 27(4): 950-959

[11] Munkres, J. (1957). "Algorithms for the Assignment and Transportation Problems." ournal of the Society for Industrial and Applied Mathematics 32-38.Ballard, D.H., "Generalizing the Hough transform to detect arbitrary shapes," Pattern Recognition 13, 2, April 1981

[12] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, Vol.13, No.2, p.111-122, 1981

[13] http://www.yokohamatire.com/smart_solution/tireology/tread_designs/

[14] D. Colbry, D. Cherba, J. Luchini "Pattern Recognition for Classification and Matching of Car Tires," submitted for presentation at the 2003 Tire Society meeting, and for consideration for publication in the journal *Tire Science and Technology*.

[15] 'Tire Guides, Inc., "Tread Assistant", www.tireguides.com/

[16] Sure Control Systems, "Wheel Inspection", http://www.sure.ca/pdf/WheelInsp.pdf, 2003.

[17] http://tireguides.com/Products/261

[18] Cunningham, P., & Delany, S.J. (2007). k-Nearest neighbour classifiers. Technical Report UCD-CSI-2007-4, Dublin: Artificial Intelligence Group.

[19] Deng-Yuan Huang, Wu-Chih Hu, Ying-Wei Wang, Ching-I Chen, Chih-Hsiang Cheng. Recognition of Tire Tread Patterns Based on Gabor Wavelets and Support Vector Machine. In Jeng-Shyang Pan, Shyi-Ming Chen, Ngoc Thanh Nguyen, editors, Computational Collective Intelligence. Technologies and Applications - Second International Conference, ICCCI 2010, Kaohsiung, Taiwan, November 10-12, 2010. Proceedings, Part III. Volume 6423 of Lecture Notes in Computer Science, pages 92-101, Springer, 2010.

[20] Moreno, P., Bernardino, A., Victor, J.S.: Gabor Parameter Selection for Local Feature Detection. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3522, pp. 11–19. Springer, Heidelberg (2005)

[21] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.

[22] Buck, U., Albertini, N., Naether, S., Thali, M.J.: 3D Documentation of Footwear Impressions and Tyre Tracks in Snow with High Resolution Optical Surface Scanning. Forensic Science International 171, 157–164 (2007)

[23] Thali, M.J., Braun, M., Brüschweiler, W., Dirnhofer, R.: Matching Tire Tracks on the Head Using Forensic Photogrammetry. Forensic Science International 113, 281–287 (2000)

[24] http://www.fosterfreeman.com/

[25] Gold, S. and A. Rangarajan (1996). "A Graduated Assignment Algorithm for Graph Matching." Transaction on Pattern Analysis and Machine Intelligence 18(4): 377-388

[26] G. Sanroma, R. Alquézar Mancho, F. Serratosa i Casanelles and B. Herrera. Smooth point-set registration using neighboring constraints. Pattern Recognition Letters, 33(15): 2029-2037, 2012.

[27] Lowe, David G. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision*. 2. pp. 1150–1157.

[28] MATLAB version 7.10.0. Natick, Massachusetts: The MathWorks Inc., 2010.